PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-038510
(43)Date of publication of application : 07.02.1995

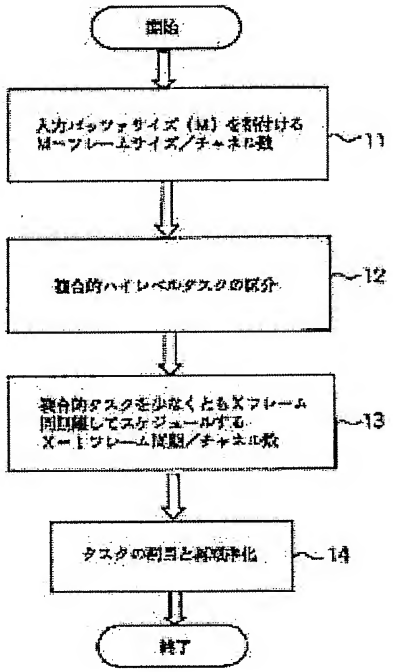
(51)Int.Cl. H04B 14/04
G06F 9/46
G11B 20/10
H03M 7/00

(21)Application number : 05-170005
(22)Date of filing : 09.07.1993

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72)Inventor : TAN PEKU YUU

(54) SCHEDULING METHOD FOR REAL-TIME TASK FOR ATRAC

(57)Abstract:
PURPOSE: To suppress the die size and energy consumption of a processor at a minimum by efficiently executing algorithms on the conditions that a lot of algorithms are applied to merchandise.
CONSTITUTION: A task scheduler is used for allocating the task of the algorithm so as to utilize calculating ability and to optimize the internal memory of a digital signal processor. Task resynchronizing time is minimized by designating the execution order of ATRAC algorithms while using the task scheduler, and the coupling of tasks and complicatedness with bottom execution time (overhead) is decreased. The task scheduler realizes efficient memory allocation for exchanging data between tasks while using a subordinary buffering method.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－38510

(43)公開日　平成７年(1995)２月７日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｈ０４Ｂ　14/04		Ｚ　4101－5K		
Ｇ０６Ｆ　9/46	３４０	Ｂ　8120－5B		
Ｇ１１Ｂ　20/10	３０１	Ｚ　7736－5D		
Ｈ０３Ｍ　7/00		8522－5Ｊ		

審査請求　未請求　請求項の数３　ＯＬ　（全 11 頁）

(21)出願番号　特願平5－170005

(22)出願日　平成５年(1993)７月９日

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　タン　ペク　ユー
シンガポール　2776　#05－305　イシュンストリート11　ブロック128

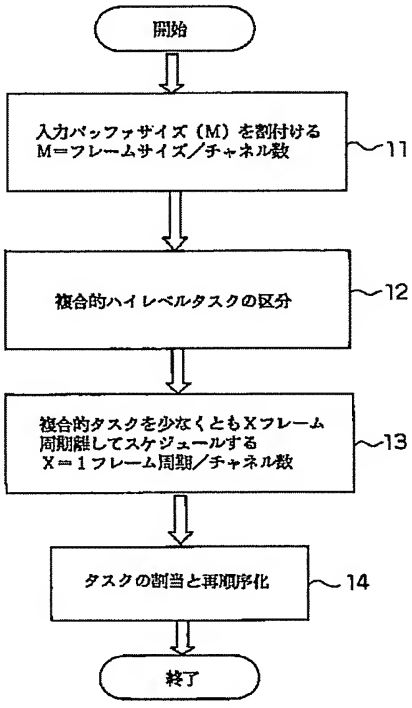
(74)代理人　弁理士　松田　正道

(54)【発明の名称】　ＡＴＲＡＣ用リアルタイム・タスクのスケジュール方法

(57)【要約】

【目的】多くの信号処理アルゴリズムが商品に適用されるようになり、アルゴリズムの効率的な実施がプロセッサのダイサイズと電力消費を最小に保つために必要となっている。

【構成】タスク・スケジューラは計算能力の利用とディジタル信号プロセッサの内部メモリを最適化するためにアルゴリズムのタスクを割り当てるために使用される。本発明では、タスク・スケジューラを使用してＡＴＲＡＣアルゴリズムの実行順序を指定してタスク再同期化時間を最小にし、最低実行付帯時間（オーバーヘッド）に伴うタスク間の結合と複雑性を減少する。タスク・スケジューラは準動的緩衝手法を使用してタスク間のデータのやり取りのための効率的メモリー割付を実現する。



【特許請求の範囲】
【請求項１】一定数の新しい入力データサンプルを使用する特定の機能を有して出力データサンプルを生成するまたはタスクの入力、出力のデータをそれぞれ制御する処理タスクをスケジュールする方法であって、
i)　タスクを指定して区割し、
ii)　プロセッサのタスクを順序化して割当て、
iii)　呼び出したタスクにメモリを割り当てるステップを備えたことを特徴とするＡＴＲＡＣ用リアルタイム・タスクのスケジュール方法。
【請求項２】同一アルゴリズムを使用した２つないしそれ以上のチャネルの処理が必要となる音声アプリケーションでのタスク間の複合性と結合を削減するスケジュール方法であって、
i)　前記フレームの［（フレーム当りのサンプル数）／（Ｎチャネル）］のバッファサイズを割当て、
ii)　２つないしそれ以上の複合し高度に結合したタスクを区割し、
iii)　少なくとも（Ｎチャネルのフレーム当りの処理時間）／Ｎ区間により後続のチャネルの処理タスクを実行し、
iv)　実現するアルゴリズムの主流部をエミュレートするためにタスクを再順序化し、タスクをホストプロセッサに割り当てる、
各ステップを備えた請求項１記載のＡＴＲＡＣ用リアルタイム・タスクのスケジュール方法。
【請求項３】準動的緩衝手法を適用することで効率的に呼び出されたタスクのバッファメモリを管理するスケジュール方法であって、
i)　呼び出されたタスクが必要とするＮ循環バッファを作成し、
ii)　Ｎ循環バッファを連結してより大きな循環バッファを形成し、
iii)　入力、出力ポインタを大きな循環バッファに指定し、
iv)　入出力ポインタをタスク実行中に更新する、
各ステップを備えた請求項１記載のＡＴＲＡＣ用リアルタイム・タスクのスケジュール方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はミニディスクで使用するディジタル信号プロセッサ（ＤＳＰ）上でＡＴＲＡＣエンコーダ・アルゴリズムを効率的に実施する、スケジュール方法に関するものである。
【０００２】
【従来の技術】通常、汎用スケジューリング手法はいくつかあり、その２つの一般的なグループとして、割り込み形および非割り込み形手法がある。非割り込み形スケジューリングは全動的、静的割当て、自己計時、全静的等の異なるスケジューリング・クラスに分けることが出来る。詳細は（エドワードA.リー及びジェフリーC.バイヤー「静的にスケジュールしたデータ流れのアーキテクチャ：ＤＳＰアプリケーション用並列アルゴリズムとアーキテクチャ」マージェィA.バヨウミ編集、pp.159 190　クルウェー・アカデミック・パブリッシャ、1991年）に詳しい。
【０００３】スケジューリング戦略の大部分ではコンパイル時間決定が必要となる。これにはタスクの実行時間を指定して行うタスク指定があり、実行条件がタスク・スケジューラの構築に必要である。上述したスケジューリング方法は複合的なタスク・スケジューリングは考慮にいれていない。
【０００４】ＤＳＰ上での信号処理アルゴリズムの実施には通常、緩衝手法が用いられ、バッファメモリを使用して先行タスクから後続タスクへ大量の処理データあるいは制御データを手渡している。更にバッファを制御するためにＤＳＰのためにコードを生成する必要がある。一般にバッファを実現し、管理する方法には、ＦＩＦＯと静的緩衝手法の２つの方法がある。ＦＩＦＯ手法は、ＦＩＦＯ待ち行列を直接使用して実現できるが、入出力ポインタを管理するのに大量の付帯の（オーバーヘッド）実行時間を必要とするのでそれらはコストが高くなる。静的緩衝手法はバッファを管理するのにかなり少ない実行時間しか必要としない。これは処理タスクの各々の呼び出しに付いて入力サンプルの位置と出力宛先の宛先を静的に計算することにより行うことが出来る。タスクを呼び出す度に、データは指定メモリ位置から直接読み取り、指定メモリ位置に直接出力する。静的緩衝手法は、タスクを呼び出す度に、呼び出したタスクはその入出力のために同一位置にアクセスすることを意味する。
【０００５】
【発明が解決しようとする課題】計算プロセスを行うため、エンコーダをディジタル信号プロセッサ（ＤＳＰ）を用いて実行する。目標はメモリ利用を最適化し、コード化のためのマシンサイクルを最小に保つことである。それによりチップ数は少なくなり、消費電力も少なくなる。しかしそれらは音質を損なわずに行うようにすべきである。
【０００６】図５はエンコーダの処理タスクの従来のスケジューリング方法を示したものである。従来のタスク割付方法は実現が容易であるが、メモリ利用と処理時間に関しては非効率的である。これはＡＴＲＡＣアルゴリズムの場合は特にそうである。各々のプロセス（図５に示すような）では、完了するのに特定の実行時間が必要である。プロセス実行時間は様々である。ＭＤＣＴ、ＱＭＦ、ビット割当プロセスといったプロセスは、他のプロセスに比べてゲインの大きなプロセスである。それらのプロセスではプロセス実行中により多くの実行時間とメモリが必要となる。図１に示す流れ図の流れにしたがってプロセスを実行する際はより多くのメモリを必要と

し、プロセス実行において非対称性をもたらす。非対称的プロセスはプロセス実行に関して余分な待機時間をもたらすことになる。

【０００７】ＭＤＣＴのようなゲインの大きなプロセスを実行するときは大きなメモリが必要となる。更に２つのＱＭＦプロセス及びＭＤＣＴプロセスは高レベルに結合されたプロセスである。ＱＭＦとＭＤＣＴ間のこの結合関係により、大きなメモリの割り付け要求は更に厳しくなる。後続のプロセスを実行する（発火する）前にスペクトルのデータを蓄積するために大きなメモリが必要となる。この状況を図６に示す。

【０００８】ＱＭＦとＭＤＣＴを処理するのに必要なメモリ要求を表１に示す。

【０００９】

【表１】

バッファ名	語
入力バッファ	2048
高ウィンドウバッファ（Ｌ／Ｒ）	1024
中間ウィンドウバッファ（Ｌ／Ｒ）	512
高ウィンドウバッファ（Ｌ／Ｒ）	512

【００１０】データ取得プロセスは背景プロセスと見なされてきた。処理時間の大きな部分は、コード化プロセスに当てられている。リアルタイム処理を行うには、２つの音声入力チャネルから16ビットデータを44.1ＫＨｚのレートで連続的に取り込む。１つは現在フレームデータ用、もう１つは次のフレームデータ用、入力バッファを複製する必要がある。これはデータ取得プロセス（背景プロセス）及びコード化プロセス（前景プロセス）の重複故に必要となる。１つのデータバッファは現在のフレーム処理に使用し、もう１つは次のフレーム処理のデータ取得に使用する。

【００１１】従って必要なメモリ容量は２音声チャネルに付き、1024 x ２ 語である。ウィンドウ化とＭＤＣＴについては、ウィンドウ・バッファサイズを表１にまとめて表現した。メモリバッファの半分はＭＤＣＴプロセスの最後にスペクトルデータを格納するのに使用し、データの他の半分は次のフレーム計算に使用する。表１は、大きなゲインプロセスとＱＭＦとＭＤＣＴ間の密接な結合により最低4096語がメモリ利用のピークであることを示している。このピークメモリ利用には、他のプロセスで必要な付帯（オーバーヘッド）メモリは含まれていない。

【００１２】大きなメモリ容量に加えて、複合的な高いレベル結合した（自然言語に近いレベルの意味である。）タスクは通常、計算集約的なタスクである。その結果、複合的なタスクでは異なるチャネルの後続タスクと再同期化するのにより多くの時間がかかることになる。複合的なタスクに続く単純なタスクを有するアルゴリズムの場合、そのタスクを通常の方法にしたがってスケジュールしようとすれば非対称処理時間を生じることになる。これはより早い処理速度を必要とする。

【００１３】静的緩衝手法の実行時間付帯時間（オーバーヘッド）は小さいが、大きなメモリ割当を必要とする。図７はタスク・スケジュールに基づく静的メモリ割当の作動を示している。各々のフレーム処理に付いて、タスクＡは３回、タスクＢは２回実行する。タスクＡが始動される度に、２ブロックのデータが生成される。合計６ブロックのデータが後続のタスクＢに対して生成される。タスクＢは各々の呼び出しに付いて３ブロックのデータを読み取る。静的緩衝手法の規則にしたがって生成されたデータと受け取った入力データは同一メモリスペースを占めるはずである。

【００１４】コード化プロセスを行うためにより多くの処理出力を有することは、コード化のために高い頻度で作動する２つないしそれ以上のＤＳＰが必要なことになる。これは電力消費とチップ数の増大をもたらす。

【００１５】メモリの非効率的な利用はＤＳＰのダイサイズが大きくなり、外部メモリのサイズの増大をもたらす。従って各々の処理を割り当ててメモリを利用する効率的な方法が必要である。この要求を満足させるため、ハードウェア実施レベルで改善を行う必要がある。計算処理方法をプロセッサに割り付け、プロセスのメモリ利用を効率的に管理する方法を開発する必要がある。

【００１６】

【課題を解決するための手段】本発明は、上記の目的を達成するため、複数音声チャネルの複合的なハイレベルで結合したタスクを解決する手段と、最適スケジューリング方法を用いてピークメモリ利用を削減する手段と、最小実行と付帯時間（オーバーヘッド）で呼び出したタスク用のバッファを管理する手段と、メモリの再利用が可能なように呼び出したタスクのためにメモリを割り当てる手段と、音質を損なうことなくＡＴＲＡＣコード化アルゴリズムの単一ＤＳＰ解決法を達成する手段とからなるリアルタイム・スケジュール方法を提供する。

【００１７】

【作用】上述のように問題を解決する方法を用いて、タ

スク・スケジューラを用いて結合した複合的な処理タスクを削減し、タスク間のバッファメモリ管理を改善する。複数タスクをリアルタイムで取り扱うタスク・スケジューラは、１つのＤＳＰ上で実行するＡＴＲＡＣアルゴリズムの異なる処理タスクを割り当てるために発明されたものである。タスク・スケジューラはメモリ利用が最適化され、処理速度が効率的に使用されるようにプロセスの実行を規定する。

【００１８】タスク・スケジューラは静的スケジュールの非割り込み形が可能となり、かつ完全に静的な特性という利点を有する。それを更に強化するため、複合的で高レベルで結合したタスクを効率的に割り当てスケジュールする方法を付け加える。非割り込み形スケジューラを使用して、プロセッサの状態を維持及び回復するのに使用する文脈スィッチングでの重大な付帯実行時間を削減する。ＡＴＲＡＣアルゴリズムの決定論的な動作の利点を考慮して、スケジュールはコンパイル時間中に決定することが出来る。ピーク時メモリ利用量を削減するため、スケジューラはＱＭＦタスクに対して0.5 フレームの入力メモリを割り当てる。スケジューラはまた、２つのタスクを各々のチャネル（フレーム当りの処理時間）／Ｎ別にスケジュールすることにより小節のサイズとＱＭＦとＭＤＣＴプロセスの結合を削減するように設計されている（Ｎはチャネル数を示す）。

【００１９】タスク間のデータのやり取りのため、タスク・スケジューラにより準動的緩衝手法を用いて呼び出したタスクにメモリを割り当てる。Ｎ循環バッファを呼び出したタスクにより作成し、より大きな循環バッファを形成するためにそのバッファを鎖状につなぐ。出力、入力ポインタはそのタスクで指定され、呼び出したタスクで更新される。Ｎ循環バッファは大きな循環バッファ内には一定のメモリ位置を持っていない。そのタスクが呼び出される度に、メモリ領域は入出力ポインタをそれに指定することにより割り当てる。ポインタを管理する際に使用する付帯実行時間（オーバーヘッド）は最小化され、これにより使用されていないメモリを他のタスクで再使用することが出来る。

【００２０】

【実施例】以下、本発明の実施例について図面を参照して説明する。図１、２、３、４は本発明の実施例を示したものである。一般にこのスケジューリング方法は処理タスクに利点があり、それは、入力データサンプル数に依存しない。この場合、ＱＭＦタスクは異なる時間間隔でスケジュール化され、１回のフレーム周期内でサンプルデータの１つのフレームの処理を完了する。このタスクは外部世界と高いレベルで結合している。プロセッサに到着したサンプルデータはリアルタイム入力する。プロセッサはサンプルデータを優先処理するために現在タスクを中断する必要がある。ハイレベル結合した複合的な処理タスクを区分して、最適バッファ利用を達成する一般化した方法を図１に示す。

【００２１】上述のタスクを使用するＮチャネル処理を行うため、（図１の参照数字11で示すように）タスクはその入力バッファに（データフレームサイズ）／（Ｎチャネル）ワードのデータを割り当てる。これにより文脈スィッチングの必要がなくなり、入力バッファサイズを削減できる。

【００２２】複合的でハイレベル結合したタスクは、同時性の利点をとるため、複数プロセッサ環境について異なるプロセッサに対しタスクの割当を実行できるように、データを区分する。しかし単一プロセッサ環境においては、ハイレベル結合した複合的なタスクは順次に実行するように指定される。これは図１で参照数字12で示す。このステップでは、複合的なタスクやハイレベルに結合したタスクは大きな小節タスクの結合を削減するように、単純なタスクに分解される。このステップを行う際、結合した複合的なタスクは単純なタスクの結合に還元される。

【００２３】単一プロセッサ環境では、複合的なタスクは異なるチャネルに対して１つのフレームサイクル内で順次にＮ回始動（実行）するようにスケジューリングされる（図１で参照数字13で示す）。このタスクは最小（１フレーム周期）／（Ｎチャネル）別にスケジュールされる。これによりチャネル間のタスクの再同期化が最小時間での実行を保証し、プロセッサに対する不必要な再同期化オーバーヘッドをなくすことが出来る。更にデータサンプルの１フレームに必要な本当の処理時間を推定するよい方法をもたらす。

【００２４】結果的には区分化より、実行アルゴリズムのタスクは再順序化と、再割当の必要を生じる。このステップはアルゴリズムの正確な流れを保証する。図１の参照数字14は、後続タスクを参照数字13で行われた複合的で高レベル結合したタスクの再スケジューリングの結果として、再順序化するするために用いられる。タスク間の待機時間を取り除くために、再割当では後続タスクの実行時間をも考慮に入れている。

【００２５】図２はこの発明を用いてＡＴＲＡＣのタスク・スケジュールを描いたものである。タスク・スケジューラは最初に（フレームサイズ／２）ワード迄の入力バッファサイズを割り付ける。

【００２６】ＱＭＦとＭＤＣＴはより複合的なタスクであり、この２つのタスクは高レベルで結合されている。

【００２７】この２つのタスクは分解されＮチャネルに対するＭＤＣＴタスクは（［フレーム当りの処理時間］／Ｎ）間隔でスケジュール化される。２チャネルのケースでは、２つのチャネルのＭＤＣＴタスクの間の時間間隔は 0.5フレーム離れている。これは図２で数字23、24で示されている。ＱＭＦタスクは数字21で示すように２チャネルあたりの0.5フレームデータを処理するために呼び出される。他の 0.5フレームのデータは、数字22で

示すように、フレームサイクルの終わりに呼び出されたＱＭＦタスクにより処理される。個々のチャネルの後続のプロセスのためのタスクは再順序化される。ブロックサイズ決定タスクは数字25で示すようにフレームの終わりにスケジュール化される。

【００２８】ＱＭＦプロセスの入力バッファサイズは上記のようにフレームサイズの 0.5である。

【００２９】図４は前記の発明を使用した結果としての１フレームサイクルに対するメモリ割当を示したものである。

【００３０】入力バッファは（フレーム当りのサンプルデータ）／（２チャネル語）のメモリ領域が割り当てられるので、入力バッファは 0.5フレームサイクル周期の後は満杯になる。

【００３１】次いて入力データは図４の数字41で示すように左右のチャネルの両方に付いてＱＭＦタスクにより処理される。

【００３２】ＱＭＦデータの出力は左右のウィンドウバッファに格納される。ＭＤＣＴタスクはその入力としてのウィンドウ（Ｈ／Ｍ／Ｌ）データを使用することによって左チャネルの処理が発火される。左チャネルに対してスケジュールされた全ての後続タスクが完了した後、ウィンドウバッファの半分を数字42で示すように他のタスクに割り当てることが出来る。

【００３３】0.5フレームサイクルが経過すると、入力バッファは満杯になり、データは数字44で示すように左チャネルのウィンドウバッファの空の半分に格納される。データ転送後、図２でスケジュールされたタスクにしたがって数字44で示すように右のチャネルに対する処理が呼び出される。左右のチャネルのスクラッチパッド・バッファとウィンドウ・バッファのメモリスペースを数字45で示すようにＭＤＣＴプロセスに続くタスク用として使用される。ＱＭＦプロセスは右チャネル処理が完了した後に始動される。ＱＭＦの出力は数字46で示すように左右のウィンドウバッファに格納される。

【００３４】準動的緩衝手法は、循環緩衝手法で使用するメモリポインタを管理により追加転送動作を削減する。

【００３５】この手法はタスク・スケジューラにより呼び出したタスクに対してメモリを割り当てのために用いられる。静的緩衝手法とは異なり、各々の入出力データフレームバッファのメモリ位置は固定されない。

【００３６】入出力ポインタはプロセッサに対して現在スケジュールされているタスクに割り当てられる。図７に示す同様な問題を用いて、図３は準動的緩衝手法を用いたメモリ割当例を示している。

【００３７】４つの循環バッファは、タスクデータ通過用の中間タスクとして、タスクＡとタスクＢにより作成される。４つのデータバッファは、より大きな循環バッファを形成するために、鎖状に結合される。循環バッファが正確に作動するには、両方のポインタは、ポインタがバッファの終わりを指したら再びバッファの初めを指し示す循環をする必要がある。図３はタスク実行終了時の循環バッファの内容を示している。時間＝０では、小さな循環バッファの４ブロックは空である。時間＝ｔ２では、タスクＡはバッファに格納する４ブロックのデータを生成し、出力バッファは更新され、ポインタは格納される。タスクＢの実行は循環バッファの３つの小さなブロックを消費する。時間＝ｔ４では、２ブロックのデータバッファがタスクＡにより生成される。後続のタスクＢは残りのブロックを消費して１フレームサイクルを完了する。動的緩衝手法の作動は、呼び出されたタスクのメモリは２フレームサイクル毎に異なるメモリ位置を占めることを示している。

【００３８】表２は、タスク・スケジューラを使用したピークメモリの利用法が、メモリ容量をかなり削減出来ることを示している。

【００３９】

【表２】

メモリバッファ	メモリサイズ（語）
入力	512
高ﾊﾞﾝﾄﾞｳｨﾝﾄﾞｳﾊﾞｯﾌｧ（左チャネル）	512
中間ウィンドウバッファ（左チャネル）	256
低ウィンドウバッファ（左チャネル）	256
高ウィンドウバッファ（右チャネル）	512
中間ウィンドウバッファ（右チャネル）	256
低ウィンドウバッファ（右チャネル）	256
スクラップ・パッド	256
出力バッファ	71（1語＝24ビット）

【００４０】入力バッファサイズは 512語に削減されている。合計2887語がピーク利用で必要となる。

【００４１】なお、本発明はオーディオ・エンコーダのサブバンド方式及び変換符号化方式でも使用することが出来る。

【００４２】

【発明の効果】以上述べたところから明らかなように、本発明はＡＴＲＡＣエンコーダの効率的なファームウエア、ハードウエア化をもたらす。

【００４３】効率的なメモリ割当及びＡＴＲＡＣアルゴリズムのタスク・スケジューリングの結果、単一のＤＳＰを用いてコード化を行うことが出来る。

【００４４】この新しいスケジューリング方法とそのメモリ管理手法は、メモリ利用及びＤＳＰの処理能力に関する点でより優れた性能をもたらす。

【００４５】実施レベルで行う改善の結果、より少ないメモリとＤＳＰのより低い作動頻度を達成することが出来る。

【００４６】また、タスク・スケジューラはまた２つのチャネルの処理タスク間の再同期化時間を最小にする。再同期化はアルゴリズムに無関係のタスクを処理するのに使用するプロセッサ時間の量、例えばタスクを呼び出す前にデータを待つのに使用する時間量として定義される。

【図面の簡単な説明】

【図１】ハイレベル結合タスクの割当てとスケジューリングのステップを示す図である。

【図２】ＡＴＲＡＣエンコードアルゴリズムのタスク・スケジュールである。

【図３】図３は準動的バッファ手法の動作である。

【図４】ＡＴＲＡＣエンコードアルゴリズムのメモリ・スケジュールである。

【図５】図５は従来技術のエンコードプロセスの基本的タスク・スケジュールである。

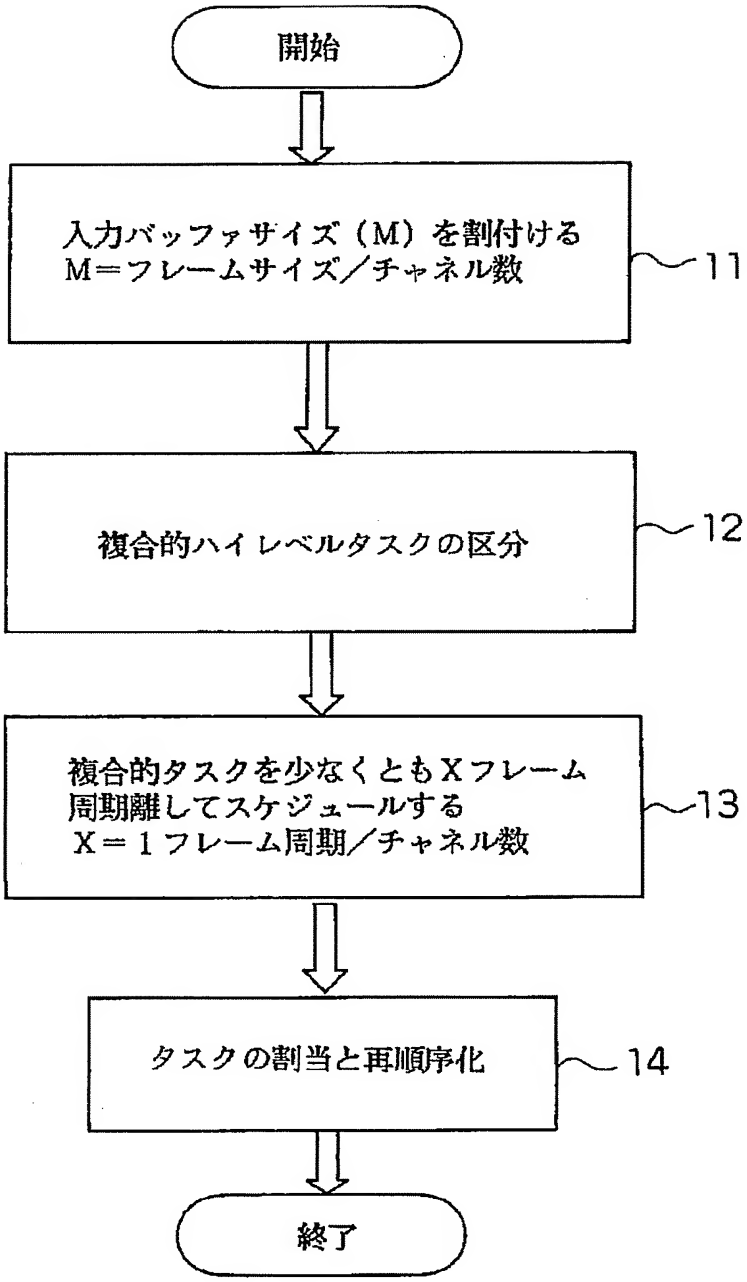
【図６】図６はＡＴＲＡＣエンコードアルゴリズムのタスクのハイレベル結合時の課題を示す図である。

【図７】図７は静的バッファを用いたメモリ割当てを示す図である。

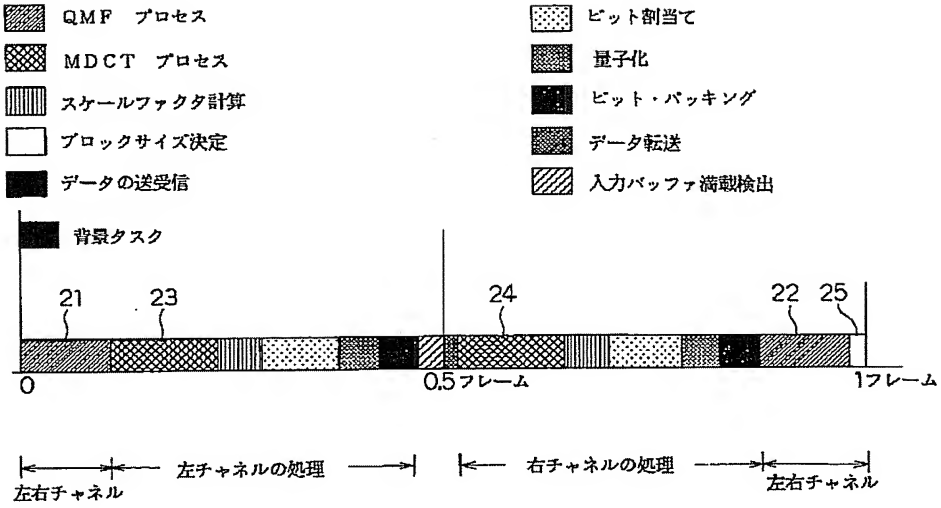
【符号の説明】

11:入力バッファサイズ（Ｍ）を割り付け
Ｍ＝フレームサイズ／チャネル数
12:複合的ハイレベル結合タスクの区分
13:複合的タスクを少なくともＸフレーム周期離してスケジュールする
Ｘ＝１フレーム周期／チャネル数
14:タスクの割当と再順序化

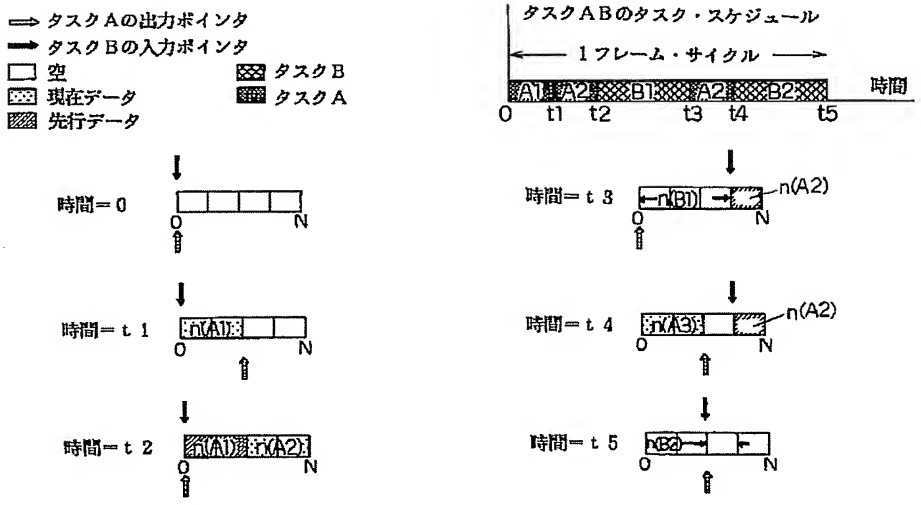
【図1】


開始
入力バッファサイズ（M）を割付ける
M＝フレームサイズ／チャネル数
11
複合的ハイレベルタスクの区分
12
複合的タスクを少なくともXフレーム
周期離してスケジュールする
X＝1フレーム周期／チャネル数
13
タスクの割当と再順序化
14
終了

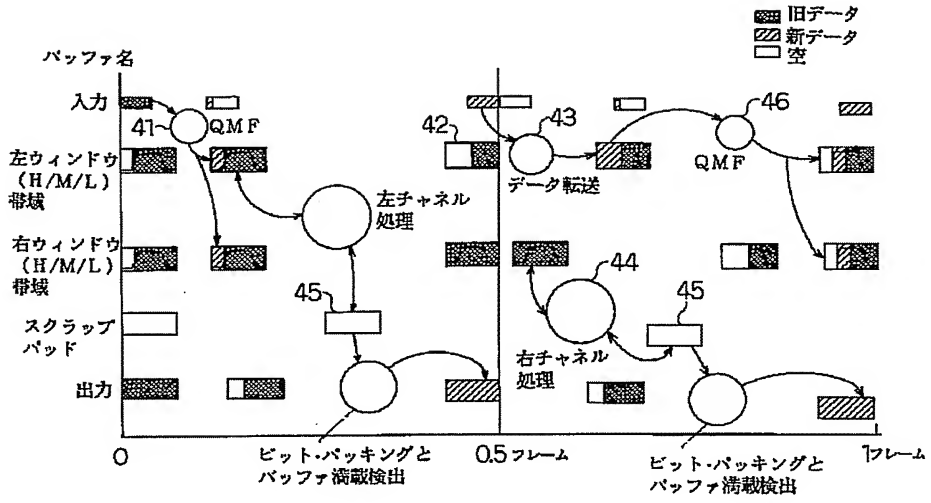
【図2】


QMF　プロセス
MDCT　プロセス
スケールファクタ計算
ブロックサイズ決定
データの送受信
ビット割当て
量子化
ビット・パッキング
データ転送
入力バッファ満載検出
背景タスク
21
23
24
22
25
0
0.5フレーム
1フレーム
左チャネルの処理
右チャネルの処理
左右チャネル
左右チャネル

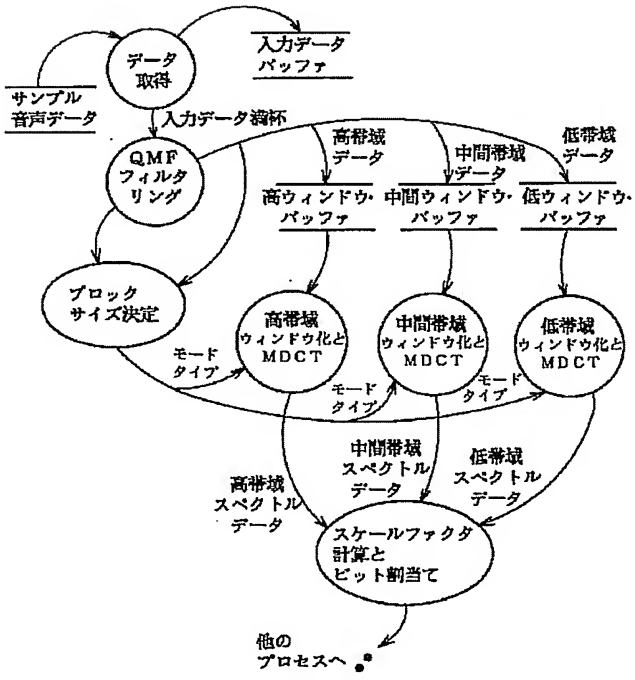
【図3】


タスクAの出力ポインタ
タスクBの入力ポインタ
空
現在データ
先行データ
タスクB
タスクA
タスクABのタスク・スケジュール
1フレーム・サイクル
時間
A1
A2
B1
A2
B2
0
t1
t2
t3
t4
t5
時間＝0
時間＝t1
時間＝t2
時間＝t3
時間＝t4
時間＝t5
n(A1)
n(A2)
n(B1)
n(A2)
n(A3)
n(A2)
n(B2)
N

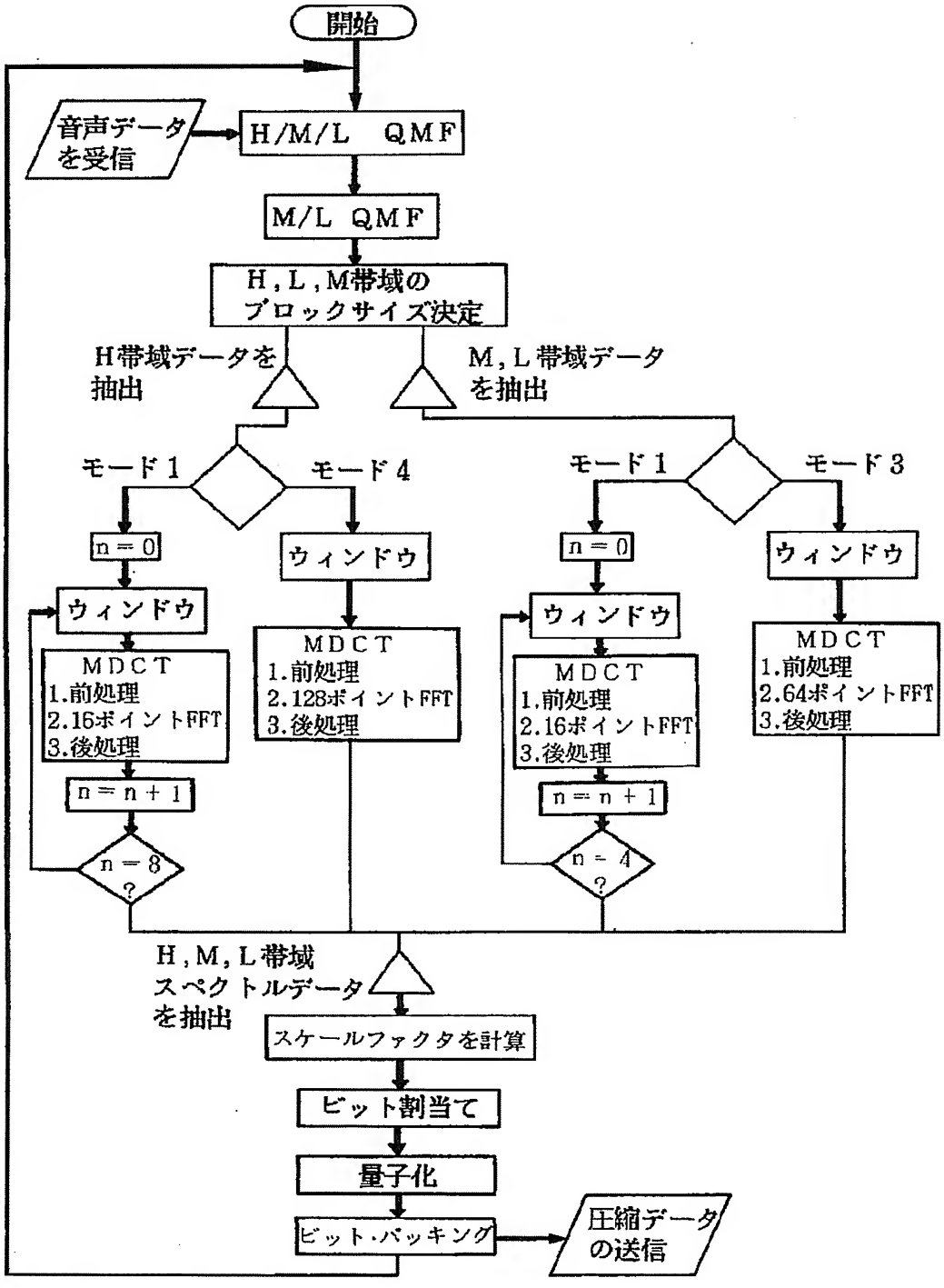
【図4】


旧データ
新データ
空
バッファ名
入力
41
QMF
42
43
46
左ウィンドウ（H/M/L）帯域
データ転送
QMF
左チャネル処理
右ウィンドウ（H/M/L）帯域
44
45
45
スクラップパッド
右チャネル処理
出力
0
ビット・パッキングとバッファ満載検出
0.5フレーム
ビット・パッキングとバッファ満載検出
1フレーム

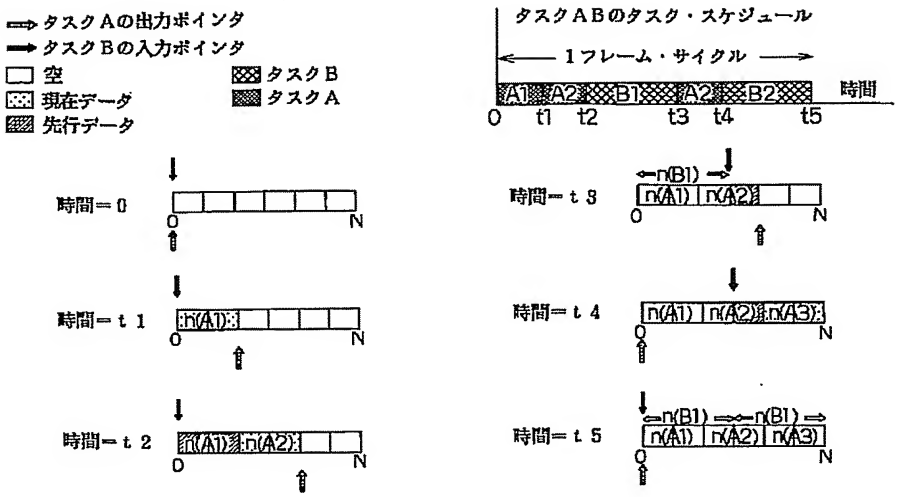
【図6】


データ取得
入力データバッファ
サンプル音声データ
入力データ満杯
QMFフィルタリング
高帯域データ
中間帯域データ
低帯域データ
高ウィンドウ・バッファ
中間ウィンドウ・バッファ
低ウィンドウ・バッファ
ブロックサイズ決定
高帯域ウィンドウ化とMDCT
中間帯域ウィンドウ化とMDCT
低帯域ウィンドウ化とMDCT
モードタイプ
モードタイプ
モードタイプ
中間帯域スペクトルデータ
低帯域スペクトルデータ
高帯域スペクトルデータ
スケールファクタ計算とビット割当て
他のプロセスへ

【図5】


開始
音声データを受信
H/M/L QMF
M/L QMF
H,L,M帯域のブロックサイズ決定
H帯域データを抽出
M,L帯域データを抽出
モード1
モード4
モード1
モード3
n=0
ウィンドウ
MDCT
1.前処理
2.16ポイントFFT
3.後処理
n=n+1
n=8?
MDCT
1.前処理
2.128ポイントFFT
3.後処理
n=0
ウィンドウ
MDCT
1.前処理
2.16ポイントFFT
3.後処理
n=n+1
n=4?
ウィンドウ
MDCT
1.前処理
2.64ポイントFFT
3.後処理
H,M,L帯域スペクトルデータを抽出
スケールファクタを計算
ビット割当て
量子化
ビット・パッキング
圧縮データの送信

【図７】


タスクＡの出力ポインタ
タスクＢの入力ポインタ
空
現在データ
先行データ
タスクＢ
タスクＡ
タスクＡＢのタスク・スケジュール
１フレーム・サイクル
A1
A2
B1
A2
B2
時間
0
t1
t2
t3
t4
t5
時間＝０
時間＝ｔ１
時間＝ｔ２
時間＝ｔ３
時間＝ｔ４
時間＝ｔ５
n(A1)
n(A2)
n(A3)
n(B1)
N